# 排序算法实验报告

## 需求分析

本实验中需要我通过设计三个算法，分别实现插入排序、快速排序、选择排序。

1. **插入排序**：即直接插入排序，它是一种十分简单的算法，具体原理即为将一个个的值插入到已经排序好的有序表中，从而形成新的有序表。抽象来说即为当我们假设前n-1（**n>=2**）个数均是已排序好的，现将第n个数插入到前面已经排序好的序列中，使得插入第n个数的序列也是排好序的。我们可以先将第n个数与最大的数比较，如果第n个数小于第n-1个数，那么则将n和n-1的值互换，并且n--，然后继续比较，直到第n个数大于第n-1个数。（n>0）
2. **快速排序**：是通过多次比较和交换来实现排序，在一趟排序中把将要排序的数据分成两个独立的部分，对这两部分进行排序使得其中一部分所有数据比另一部分都要小，然后继续递归排序这两部分，最终实现所有数据有序。首先设置一个分界值也就是基准值又是也称为监视哨，通过该分界值将数据分割成两部分。将大于或等于分界值的数据集中到右边，小于分界值的数据集中到左边。一趟排序过后，左边部分中各个数据元素都小于分界值，而右边部分中各数据元素都大于或等于分界值，且右边部分个数据元素皆大于左边所有数据元素。然后，左边和右边的数据可以看成两组不同的部分，重复上述步骤，当左右两部分都有序时，整个数据就完成了排序。
3. **选择排序**：选择排序其实和冒泡排序比较类似，即为双重遍历数组，每一轮都经过比较分析找到最小元素的下标，然后将它交换到首位，通过这样达到排序的目的。

## 概要设计

首先我定义了一个print(int arr[10],int n)函数，用于输出排序好后的数组。

然后定义InsertionSort(int arr[10],int n)，通过j、k、temp等的反复对比实现插入排序。

然后定义QuickSort(int arr[10],int l,int n)，通过first、last来指代数组元素，然后运用l来区分上下界，最后实现快速排序

定义SelectionSort(int arr[10],int n)，通过temp和k来实现选择排序。

在main函数中首先定义三个数组arr1、2、3，然后先用for函数循环读入10个数到arr数组中，再复制到arr2,3中，以达到三个函数中每次排序的初始效果都相同且排序结果互不干扰的目的。

## 详细设计

void print(int arr[10],int n){

for(int i=0;i<n;i++){

printf("%d ",arr[i]);

}

printf("\n");

}

print函数，通过给出数组arr[10]和总数n，使用for循环依次输出数组的值。在最后输出完成后再打印换行符。

void InsertionSort(int arr[10],int n){

int j,temp;

for(int i=1;i<n;i++){

j=i-1;

int k=i;

while(k>=0&&j>=0&&arr[k]<arr[j]){

temp=arr[k];

arr[k]=arr[j];

arr[j]=temp;

j--;

k--;

}

}

}

插入排序，首先指明整数类型的数j和temp，运用for循环，在1~n上循环遍历arr[10]内的元素，对于每次循环内，都会初始赋值j=i-1和k=i。

由于我们的算法是按从小到大来排序的，故当我们想要插入第i个元素时（代码中的第k个元素，等值），我们首先需要 将其与arr中的第j个元素比较，即与第i-1个元素比较，如果arr[k]>arr[j]，则说明第i个元素比第i-1个元素还要大，那么则不需要插入前面的位置，可以直接顺接在n-1个元素后面。

如果arr[k]<arr[j]，那么我们现在需要temp值来进行arr[k]和arr[j]元素值的互换，在互换arr[k]和arr[j]的值后，我们将k--,j--，现在我们需要比较arr[i-1]和arr[i-2]的值的大小，即为进行k--,j--后的arr[k]和arr[j]的大小，如果arr[k]依然小于arr[j]那么我们依然需要互换值，直到arr[k]>arr[j]或者已经调换到最首位。

void QuickSort(int arr[10],int l,int n){

if(l+1>n){

return;

}

int first=l,last=n-1,key=arr[first];

while(first<last){

while(first<last&&arr[last]>=key){

last--;

}

//如果值小于 key分界值 交换

arr[first]=arr[last];

while(first<last&&arr[first]<key){

first++;

}

//如果值大于key分界值 交换

arr[last]=arr[first];

}

arr[first]=key;

//递归左右部分进行快排

QuickSort(arr,l,first);

QuickSort(arr,first+1,n);

}

快速排序：首先设置三个参数，first指向区间左端，last指向区间右端，key为当前的分界值。从待排序的数据元素中选取一个通常为第一个作为基准值元素（key）key=arr[0]，设置双指针first指向区间左端，last指向区间右端。key 首先与 arr[last] 进行比较，如果 arr[last]<key，则arr[first]=arr[last]将这个比key小的数放到左边去，如果arr[last]>=key则- -last，再拿arr[last]与key进行比较，直到arr[last]<key交换元素为止。arr[last]<key交换元素后，转向左边部分，用arr[first]与key进行比较，如果arr[first]<key,则++first，然后继续进行比较，直至arr[first]>=key,则将arr[last]=arr[first]。

void SelectionSort(int arr[10],int n){

int temp,k;

for(int i=0;i<10;i++){

temp=i;

for(int j=i;j<10;j++){

if (arr[temp]>arr[j]){

temp=j;

}

}

k=arr[i];

arr[i]=arr[temp];

arr[temp]=k;

}

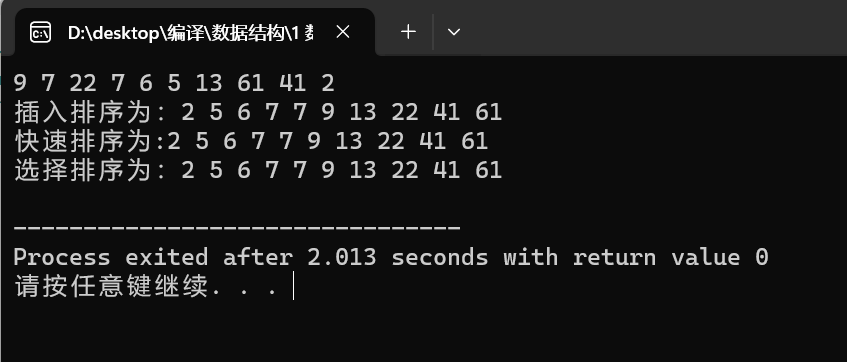
}

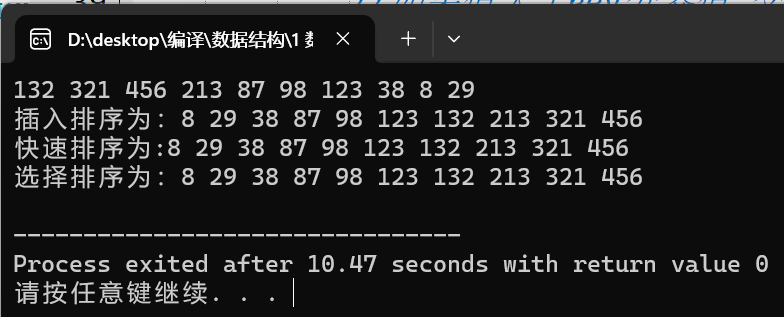
选择排序函数：定义了一个整数型数据temp和k，先用一层for循环，定义temp为i，即定义最小的数的坐标为i，然后再嵌套一层for循环，先定义j=i，表示从i开始选择，因为我们已经定义i之前的是排序好的且最小的几个数，然后如果我们找到j使得arr[j]<arr[temp]，那么就令temp=j，在j的for循环结束后，我们利用k来将arr[i]和arr[temp]的值互换，达到排序的效果。

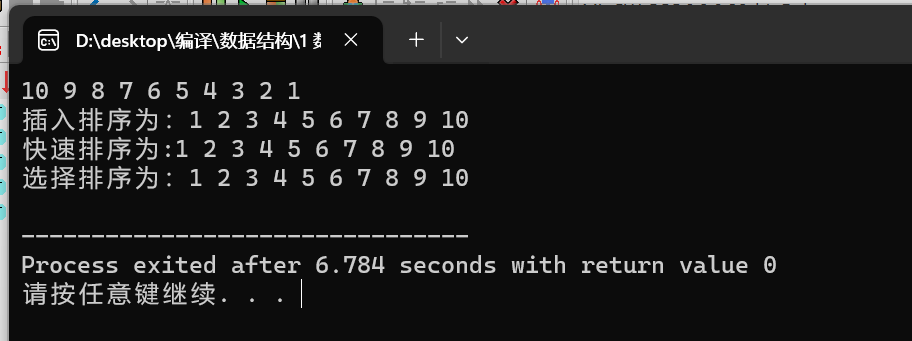
## 调试分析

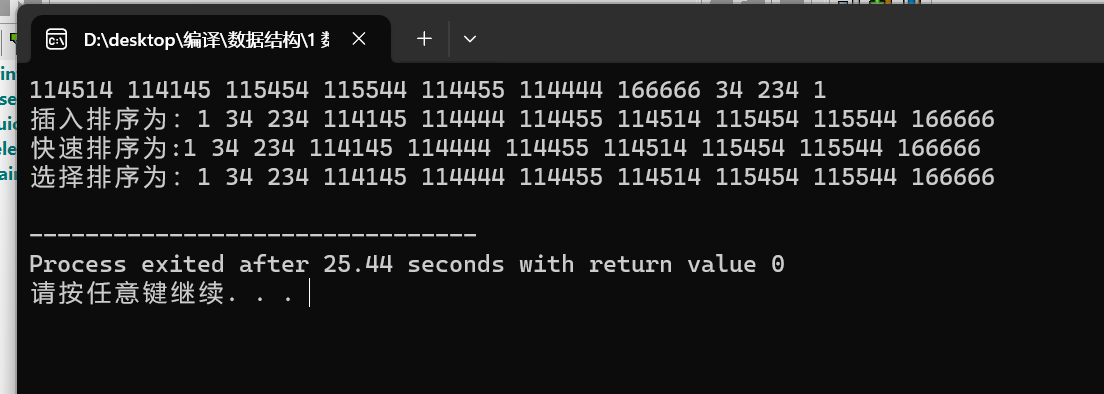
在编辑实验中我遇到了很多问题，其中最突出的问题就是在选择排序的过程中，由于我前两个算法的编辑里的temp都指代arr数组里的元素值，而第三个函数中的temp=i，因为第三个函数里的temp其实质是数组中当前最小值的坐标，而我最开始写成了temp=arr[i]，这导致当我输入测试样例后输出的答案完全不对。在我仔细的调试后，终于找到了问题之所在，解决了问题。

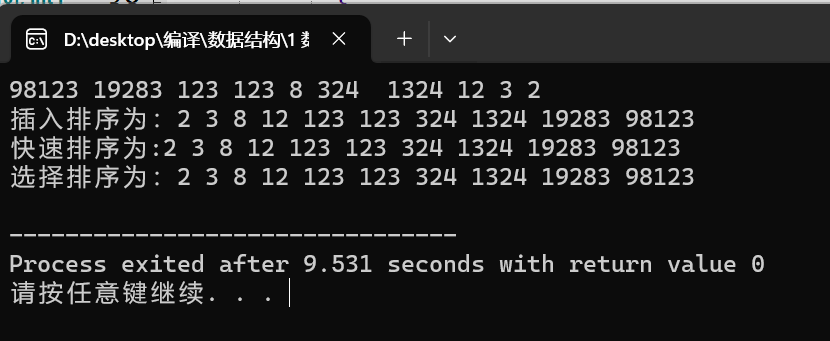
## 测试结果











## 附录

#include <stdio.h>

void print(int arr[10],int n){

for(int i=0;i<n;i++){

printf("%d ",arr[i]);

}

printf("\n");

}

void InsertionSort(int arr[10],int n){

int j,temp;

for(int i=1;i<n;i++){

j=i-1;

int k=i;

while(k>=0&&j>=0&&arr[k]<arr[j]){

temp=arr[k];

arr[k]=arr[j];

arr[j]=temp;

j--;

k--;

}

}

}

void QuickSort(int arr[10],int l,int n){

if(l+1>n){

return;

}

int first=l,last=n-1,key=arr[first];

while(first<last){

while(first<last&&arr[last]>=key){

last--;

}

//如果值小于 key分界值 交换

arr[first]=arr[last];

while(first<last&&arr[first]<key){

first++;

}

//如果值大于key分界值 交换

arr[last]=arr[first];

}

arr[first]=key;

//递归左右部分进行快排

QuickSort(arr,l,first);

QuickSort(arr,first+1,n);

}

void SelectionSort(int arr[10],int n){

int temp,k;

for(int i=0;i<10;i++){

temp=i;

for(int j=i;j<10;j++){

if (arr[temp]>arr[j]){

temp=j;

}

}

k=arr[i];

arr[i]=arr[temp];

arr[temp]=k;

}

}

int main(){

int arr1[10],arr2[10],arr3[10];

for(int i=0;i<10;i++){

scanf("%d",&arr1[i]);

arr2[i]=arr1[i];

arr3[i]=arr1[i];

}

//插入排序

InsertionSort(arr1,10);

printf("插入排序为：");

print(arr1,10);

//快速排序

printf("快速排序为:");

QuickSort(arr2,0,10);

print(arr2,10);

//选择排序

printf("选择排序为：");

SelectionSort(arr3,10);

print(arr3,10);

return 0;

}